Alarm Bit Control

Local application

Robert Goodwin

Thu, Dec 2, 2010

The normal front end alarm scanning support allows asserting a control line when any channel of a specified subset is in the BAD state; *i.e.*, out of limits. But there can be cases when more than one control line needs asserted depending on which set of channels is BAD. This note describes a new local application called BADC that is designed to provide this option.

Overall scheme

Monitor the alarm flags for a set of analog channels to see if any is currently in the BAD state, as noted during the last cycle's alarm scan. If so, assert a control line to mark this fact; otherwise, de-assert that same line. Alarm scanning is done at 15 Hz, so the control line can track well.

The parameter layout is as follows:

Param		*Size*	*Meaning*
ENABLE	B	2	Usual LA enable Bit#
CONTROL	B	2	Output control Bit#
CHANS	C	2	Chan# parameters indicating which channels to monitor
. . .			

Up to 8 channel number parameters can be specified. Any channel# parameter of zero is ignored. If consecutive channels are to be monitored, a range can be specified using 2 words in which the second word has its sign bit set. For example, to monitor channels `0x0120–0127`, specify the two consecutive parameters `0x0120`, `0x8127`. Except for this case, the channels to be monitored can be specified in any order. The total number of channels is currently limited to 136, an arbitrary choice.

The CONTROL parameter specifies the Bit# to be set or cleared, according to the result of the scan. The sign bit of this Bit# parameter specifies the state of the control line to be set when one or more of the specified channels is in the BAD state.

To set more than one control line, merely use a separate instance of the LA for each line needed. Each instance can thus specify a different set of channels to be monitored.

The LA watches for channel# parameters to be modified, in which case it re-issues a data request for the new set of specified channels.

Caveat

Each cycle, the alarm scan is performed *after* the local applications run. A local application monitoring the alarm flags therefore sees their state on the previous cycle, since the alarm scan has not yet run on the present cycle. The control line state is therefore one cycle behind.

Implementation details

A data request is made for the alarm flags for the entire set of channels specified via the channel number parameters. Each cycle, call for this data, and loop through all alarm flags looking for cases in which both the Active bit ACT and the BAD bit are set, implying that the channel is currently being scanned for alarms, and its current state is out of limits, or BAD. If any such case is seen, declare the current composite state of this channel set to be BAD; otherwise, it is GOOD. If it is BAD, then the control Bit is set to its BAD state, whose polarity is given by the sign bit of the CONTROL Bit# parameter; otherwise, it is set to the opposite polarity. Only make the setting of this DC control line when its state needs to be changed.

Trip log

An internal trip log is maintained for diagnostic monitoring. An entry is made each time the control line is asserted. When it is subsequently de-asserted, this same entry is updated with the duration, in 15 Hz cycles, that the control line was asserted. The 8-byte entry format is as follows:

Field	*Size*	*Meaning*
`TIME`	6	BCD time-of-day, as Mo, Da, Hr, Mn, Sc, Cy
`CYCLES`	2	#cycles duration of the control line, limited to 32767, or ~36 minutes.

The circular log is currently sized for 64 entries and is located at static memory offset `0x0140`.

Here is an annotated example of such log entries via the Print Memory page application:

```
FILE<0647>  BADC+140   12/02/10 1407
 0647:00072946   1202 0935 4113 01D3      12/02 0935:41-13, 467 cycles = 31.1 sec
     :0007294E   1202 0951 4506 0026      12/02 0951:45-06,  38 cycles =  2.5 sec
     :00072956   1202 1126 4810 0062      12/02 1126:48-10,  98 cycles =  6.5 sec
     :0007295E   1202 1152 4812 009A      12/02 1152:48-12, 154 cycles = 10.3 sec
     :00072966   1202 1209 4711 0002      12/02 1209:47-11,   2 cycles =  0.1 sec
     :0007296E   1202 1232 3300 0023      12/02 1232:33-00,  35 cycles =  2.3 sec
     :00072976   1202 1246 2605 001F      12/02 1246:26-05,  31 cycles =  2.1 sec
     :0007297E   1202 1252 5711 0021      12/02 1252:57-11,  33 cycles =  2.2 sec
```